PCT

WORLD INTELLI

WO 9604604A1
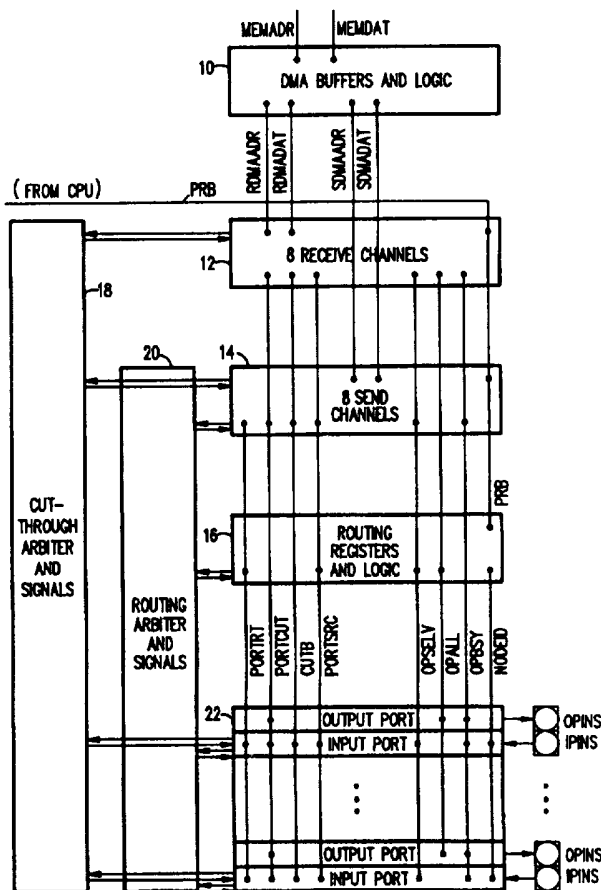
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<table>
<tr><td>(51) International Patent Classification ⁶ :

G06F 15/16</td><td>A1</td><td>(11) International Publication Number: WO 96/04604

(43) International Publication Date: 15 February 1996 (15.02.96)</td></tr>
</table>

(21) International Application Number: PCT/US95/09474

(22) International Filing Date: 28 July 1995 (28.07.95)

(30) Priority Data:
    08/283,572   1 August 1994 (01.08.94)   US

(71) Applicant *(for all designated States except US)*: NCUBE [US/US]; 110 Marsh Drive, Foster City, CA 94404-1184 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: DUZETT, Robert, C. [US/US]; 522 N.E. Geraldine Drive, Hillsboro, OR 97124 (US). KENOYER, Stanley, P. [US/US]; Route 1, Box 199, Forest Grove, OR 97116 (US).

(74) Agents: YOUNG, Barry, N. et al.; Gray Cary Ware & Freidenrich, 400 Hamilton Avenue, Palo Alto, CA 94301-1825 (US).

(54) Title: NETWORK COMMUNICATION UNIT USING AN ADAPTIVE ROUTER

(57) Abstract

A parallel processor network comprised of a plurality of nodes, each node including a processor containing a number of I/O ports, and a local memory. A communication path is established through a node by comparing a target node address in a first address packet with a processor ID of the node. If node address is equal to the target node address a receive channel is allocated to the input port and a route ready command is sent over an output port paired with the input port. If the node address is not equal to the target node address, then a first unallocated output port is selected from a port vector and the address packet is forwarded to a next node over the selected output port.

5   NETWORK COMMUNICATION UNIT USING AN ADAPTIVE ROUTER

Cross-reference to Related Application

Copending application serial number 07/587,237 entitled "Network Communication Unit for use

in a High Performance Computer System" of Stephen R. Colley, et al., filed on September 24,

10   1990, assigned to nCUBE Corporation, the assignee of the present invention, and incorporated

herein by reference.

Background of the Invention

15   Field of the Invention

The invention relates to data-processing systems, and more particularly, to a communication

mechanism for use in a high-performance, parallel-processing system.

Description of the Prior Art

20   US patent 5,113,523 describes a parallel processor comprised of a plurality of processing

nodes, each node including a processor and a memory. Each processor includes means for

executing instructions, logic connected to the memory for interfacing the processor with the

memory and an internode communication mechanism. The internode communication

mechanism connects the nodes to form a first array of order n having a hypercube topology. A

25   second array of order n having nodes connected together in a hypercube topology is

interconnected with the first array to form an order n+1 array. The order n+1 array is made

up of the first and second arrays of order n, such that a parallel processor system may be

structured with any number of processors that is a power of two. A set of I/O processors are

connected to the nodes of the arrays by means of I/O channels. The internode communication

30   comprises a serial data channel driven by a clock that is common to all of the nodes.

The above-referenced Copending application SN 07/587,237, describes a fixed-routing

communication system in which each of the processors in the network described in US patent

5,113,523 is assigned a unique processor identification (ID). The processor IDs of two

35   processors connected to each other through port number n, vary only in the nth bit. A plurality

-2-

of input ports and a plurality of output ports are provided at each node. Control means at one
of the input ports of the node receives address packets related to a current message from an
output port of another of the nodes. A data bus connects the input and output ports of the
node together such that a message received on any one input port is routed to any other

5   output port. A compare logic compares a node address in a first address packet with the
processor ID of the node to determine the bit position of the first difference between the node
address in the first address packet and the processor ID of the node. The compare logic
includes means for activating for transmission of the message packet placed on the data bus
by the input port, the one of the plurality of output ports whose port number corresponds to

10   the bit position of the first difference, starting at bit $n+1$, where n is the number of the port on
which the message was received.

In the fixed routing scheme described in the above-referenced application SN 07/587,237, a
message from a given source to a given destination can take exactly one routing path, unless it

15   is forwarded, cutting through intermediate nodes and blocking on busy channels until the path
is established. The path taken is the dimension-order minimum-length path. While this scheme
is deadlock-free, it will not reroute messages around blocked or faulty nodes.

It is an object of the present invention to provide a new communication mechanism that will

20   route messages around blocked or faulty nodes in a parallel processor.

Summary of the Invention
Briefly, the above object is accomplished in accordance with an embodiment of the present
invention by providing a maze adaptive routing mechanism. In a maze adaptive routing scheme

25   for a transmission from node A to node B, all minimum-length paths between the two nodes
are searched by a single-packet scout that attempts to find a free path.

One minimum path at a time is scouted, starting with the lowest-order uphill path and doing a
depth-first, helical traversal of the minimum-path graph until a free path to the destination is

30   found. If no free minimum-length path is found, other, non-minimum-length paths may be

-3-

searched; or the central processing unit is interrupted so that software can restart the search or implement some other policy.

5   In accordance with an aspect of the invention, a node address packet, used for finding and establishing a route to a destination node, is provided with destination (target) node address, plus other necessary control bits and parameters.

The invention has the advantage that the mechanism automatically routes around blocked or disabled nodes.

10

The maze router also exhibits superior bandwidth usage and latency for most message mixes. This is attributed to its exhaustive yet sequential approach to route searching. The maze router eliminates the blockage of the fixed routing wormhole scheme, yet keeps route-search traffic to a minimum.

15

Brief Description of the Drawings

The foregoing and other objects, features, and advantages of the invention will be apparent from the following detailed description of a preferred embodiment of the invention, as illustrated in the accompanying drawings wherein:

20

FIGURE 1 is a detailed block diagram of a communications unit in which the present invention is embodied;

FIGURE 2 is block diagram of a receive channel shown in FIGURE 1;

FIGURE 3 is block diagram of a send channel shown in FIGURE 1;

25   FIGURE 4 is block diagram of an input port shown in FIGURE 1;

FIGURE 5 is block diagram of an output port shown in FIGURE 1;

FIGURE 6 is block diagram of routing registers and logic shown in FIGURE 1;

FIGURE 7 is a maze routing example in an eight processor network dimension 3 hyper cube;

Figure 8 is a graph of message latency versus percent of network transmitting data; for fixed

30   and adaptive routing.

Figure 9 is a diagram of an address packet;

-4-

Figure 10 is a diagram of a data packet;

FIGURE 11 is a diagram of a command packet;

FIGURE 12 is a flow diagram of the routing state of a send operation;

FIGURE 13 is a flow diagram of the routing state of an input port operation;

5 FIGURE 14 illustrates end-to-end acknowledge;

FIGURE 15 is a maze-route timing diagram wherein there are no blocked links; and,

FIGURE 16 is a maze-route timing diagram wherein there are blocked links.

DESCRIPTION OF THE PREFERRED EMBODIMENT

10 Signal Line Definitions

The following is a summary of signal line abbreviations used in FIGURE 1 and their definitions:

CPU- Central Processing Unit.

CUTB- cut through bus by which commands, addresses and data are passed among ports and channels.

15 IPINS- input pins.

MEMADR- Memory address bus.

MEMDAT- Memory data bus.

NODEID- Node identification -a unique code number assigned to each node processor to distinguish a node from other nodes.

20 OPALL- output port allocation, one for each of 18 ports and 8 receive channels.

OPBSY- Output port busy -one line for each of 18 output ports and 8 receive channels to indicate that the corresponding output port or channel is busy.

OPINS- output pins.

OPSELV- output port and receive channel select vector; routing logic indicates the output port

25 or channel selected.

PORTCUT- a vector indicating to which port, send channel or receive channel to cut through.

PORTSRC- a vector indicating which input port or send channel requests a route.

PORTRT- a vector of candidate ports from an agent requesting a route.

PRB- Processor Bus - a data bus from the central processing unit (CPU).

30 RDMAADR- receive DMA address.

RDMADAT- receive DMA data.

-5-

SDMAADR- send DMA address.

SDMADAT- send DMA data.

Command Definitions

5   ETE-ack- End-to -end acknowledge - when a transmission has completed, backtracking takes

place in the reverse direction along the transmission route as ETE-ack logic retraces the path in

order to deallocate all ports in that path and delivers status to the originating send channel.

BOT- Beginning of Transmission - A signal generated by a send instruction to a sending

10   channel that indicates the beginning of a transmission.

EOM -End of message is a command delivered to the target node that indicates that this is the

end of a message.

15   EOT -End of transmission is a command delivered to the target node that indicates that this is

the last packet of this transmission.

ETE_ack -The End-to -end acknowledge command indicates a transmission was delivered

successfully to the target node. It includes a receive code set up by the software at the

20   receiver end.

ETE_nack -The End-to -end not acknowledge command indicates that a message was not

delivered successfully to the target node and returns status such as parity error or receive

count overflow.

25

ETE_en -The end-to-end enable signal is sent with the address packet to indicate that the end-

to-end logic at the receive channel in the target node is enabled.

Flush_path - is a flush command that deallocates and frees up all ports and channels in a path

30   to a target node.

Reset_node - Reset_ node is a command that resets a node and its ports and channels to an initial state.

Reset_CPU - Reset_CPU is a command that resets a CPU at a node but not its ports and
5   channels to an initial state.

Rcv_rej - Receive reject is a path rejection command that indicates that no receive channel is available at the target node.

10   Rcv_rdy -Receive ready is a command that indicates that the receive channel is ready to accept a transmission.

Route_rdy -Route ready is a command that indicates to a send channel that a requested route to the target node and receive channel has been found and allocated for a transmission from
15   the send channel.

Route_reject -Route reject is a path rejection command that indicates that all attempted paths to the target node are blocked.

20   Rt_ack - Route acknowledge is a path acknowledge command that indicates that the path that a scout packet took to the target node is available and allocated (reserved).

Send_rdy -Send ready is a status register that indicates the send channels that are ready to start a message or transmission.
25

Refer to **FIGURE 1** which is a detailed block diagram of a communications unit in which the present invention is embodied. A direct memory access (DMA) buffers and logic block (10) is connected to a main memory (not shown) via memory address (memadr) and memory data (memdat) buses. Eight receive (rcv) channels (12) are paired with eight send channels (14). A
30   routing registers and logic block (16) is connected to the send channels via Portrt and Opselv, and to the receive channels via Opselv. A cut-through arbiter and signals block (18) is

-7-

connected to the send and receive channels. A routing arbiter and signals block (20) is
connected to the send channels (14) and to the routing registers and logic (16). Eighteen
output port/input-port pairs (22) are connected to the cut-through arbiter and signals (18), the
routing arbiter and signals (20), and to the send and receive channels via Portcut, Cutb, and

5   Portsrc. A processor bus (prb) is connected from a central processing unit (not shown) to the
receive channels, the send channels, the routing registers and logic and to input and output
ports (22). The input ports are connected to the routing arbiter and signals block (20) and off-
chip via the input pins (Ipins). The output ports are connected to the cut-through arbiter and
signals block (18) and off-chip via the output pins (Opins).

10

Receive Channel
Refer to **FIGURE 2** which is block diagram of one of eight receive channels (12) shown in
FIGURE 1. Each receive channel includes a receive direct memory access register, RDMA,
(50), a receive status register, RSTAT, (52), a DMA four word-deep buffer DMABUF, (54) and

15   a receive source vector register, RSRCVEC, (56). The input port that cuts through transmission
data to this receive channel is indicated by the contents of the RSRCVEC register; which is
placed on the PORTCUT bus when the receive channel returns an ete-ack or ete-nak command.

An address packet or data packet is received from an input port over the cut through bus

20   CUTB. Data are buffered in the receive DMA buffer DMA BUF(54) before being written to
memory. An address and length to describe where in memory to place the data is stored in the
receive dma register (50). As data is received it is transferred over the data write bus DWB
to a memory controller along with the address DADR and word count DCNT.
The receive source vector register RSRCVEC is an indication of from which input port the data

25   was sent. An end to end (ETE) command, with end-to end (ETE) status, is sent back to the
sending port from the RSTAT register (52).

Send Channel
Refer to **FIGURE 3** which is block diagram of one of eight send channels (14) shown in

30   **FIGURE 1**. Each send channel includes a send buffer DMA, SBDMA (58), send DMA register,
SMDA (60), send buffer path, SBPTH (62), send path register, SPTH (64), end-to-end buffer,

-8-

ETEB (66), end-to-end register, ETE (68), DMA buffer, DMABUF (70), send port vector register, SPORTVEC (72), send port select register, SPORTSEL (74) and send port alternate register, SPORTALT (76).

5   The SDMA register (60) stores address and length fields, double queued with the SBDMA register (58). The SPTH register (64) stores port vector and node address of the destination (target) node, double queued with the SBPTH register (62). At the end of a transmission the SBDMA register (58) is popped to the SDMA register (60) and the SBPTH register (62) is popped to the SPTH register (64). At the end of a message the SBDMA only is popped, to the

10   SDMA register.

       The ETE register (68) is the top of the end-to-end (ETE) queue and the ETEB register (66) is the bottom of the end-to-end (ETE) queue. An end-to-end ETE is returned via the CUTBUS and stored in the ETEB register (66). The ETEB is popped to the ETE if the ETE is empty or invalid.

15
       CPU Instructions access the registers by means of the PRB bus. If a SDMA register is in use, the information is placed in the buffer SBDMA(58). If SBDMA (58) is also full, a flag is set in the CPU indicating that the send channel is full. (same for SPTH and SBPTH)

20   The send port vector SPORTVEC (72) is in a separate path in the send channel but is part of the SPTH register. SPORTVEC stores a bit pattern indicating through which output ports the transmission may be routed. The port vector is passed by means of the port route PORTRT bus to the routing logic shown in **FIGURE 6**. A PORTSRC line is asserted to indicate which channel or port is requesting the new route. If accepted, a vector, which is a one in a field of

25   zeros, is sent from the routing logic via the output port select OPSEL bus to the send port select SPORTSEL register (74). The SPORTSEL register indicates the one selected output port. The send channel sends an address packet, and/or data from the DMABUF (70), to the output port via the cut through bus CUTB. The output port for the cut through is selected by placing the port select vector SPORTSEL (74) on the port cut-through select bus PORTCUT.

30

-9-

The SPORTVEC vector is inverted and the inverted vector is placed in the alternate port vector register SPORTALT (76). If all attempted routes fail using the SPORTSEL register, the SPORTALT register is transferred to the SPORTSEL to provide an alternate route select attempt.

5   The output port allocated OPALL lines, are activated to prevent another channel or port from interfering with the selected (allocated) port.

Input Port

Refer to **FIGURE 4** which is block diagram of an input port shown in **FIGURE 1**. Each input
10   port includes input data register, IDAT (78), input data buffer, IBUFDAT (80), input buffer command, IBUFCMD (81), input back track data register, IBAKDAT (82), identification difference register, IDDIF (84), input port source register, IPORTSRC (86), input port vector register, IPORTVEC, (88) and input port select register, IPORTSEL (90).

15   Pairs of bits that are shifted out of an output port of a corresponding hypercube neighbor node are shifted into the input port command IBUFCMD register (81) on the IPINS. At the front of a packet is the packet type which indicates the size of the packet. If a short packet, it is shifted into the middle of the IBUFDAT register (80). If it is backtrack command, the bits are shifted into the IBAKDAT register (82). If an address is shifted in, it is compared with an address of
20   the node, NODEID. The result is a difference vector that is loaded into the IDDIF register (84). If folding is enabled, the FOLDEN line is asserted and the ID bits corresponding to the folded port are used to modify the difference vector IDDIF accordingly. The contents of the IDDIF register are loaded into the input port vector register IPORTVEC (88) which is used to identify the next minimum path ports through which a message may be routed. The IPORTVEC is sent to the
25   routing logic of **FIGURE 6** via the port route bus PORTRT. At the same time, the input port asserts its corresponding bit on PORTSRC, which is passed to the routing logic with port routing.

The output port selected by the routing logic is indicated with the OPSELV bus, which is
30   written into the iportsel register. Also, the PORTSRC value is written into the srcvec register of the iport corresponding to the selected oport. If a back track command is received at an

-10-

iport via the IBAKDAT register, SRCVEC selects the oport to which the back track data is sent. Any time data is put out on the CUTB bus, the contents of the IPORTSEL register (90) or of the SRCVEC register are put out on the PORTCUT bus to select the output port to receive the data.

5

Output Port

Refer to **FIGURE 5** which is block diagram of an output port shown in **FIGURE 1**. Each output port includes output data register, ODAT (92), output data buffer, OBUFDAT (94), output backtrack data register, OBAKDAT (96) and output acknowledge data register, OACKDAT (98).

10

An address or data packet arrives from an input port or send channel on the cut through bus CUTB and is loaded into the output data ODAT register (92). The ODAT register is popped into the output buffer data OBUFDAT register (94) if it is not busy. If ODAT is full, an output port busy OPBSY line is asserted. The output backtrack data OBAKDAT register (96) stores

15 backtrack commands. The output acknowledge data OACKDAT register (98) stores packet acknowledge commands. OBUFDAT register (94), OBAKDAT register (96), and OACKDAT register (98) are shift registers that shift bits out of the OPINS every clock period, two pins (bits) per clock period.

20 ## Routing Registers and Logic

Refer to **FIGURE 6** which is block diagram of the routing registers and logic (16) shown in **FIGURE 1**. The routing registers include node identification register, NODEID (100), termination register, TERMIN (102), fold enable register, FOLDEN (104), output port allocation register, OPALL (108), output port busy register, OPBSY (110), alternate mask register, ALTMSK (112), input

25 output mask, IOMSK (114), input output select, IOSEL (116), OPORTEN (118) and a routing find first one (FFO) logic(120).

The NODEID register (100) contains the node address of this processor.

-11-

The terminal register TERMIN (102) indicates which input ports are terminal ports. Terminal ports do not compare address packets with the NODEID and any address packet that arrives at such a port is accepted as having arrived at the target node.

5   The fold enable register FOLDEN (104) holds a vector which indicates which ports can be folded. FOLDEN is considered by the routing FFO (120) when performing a wormhole routing protocol, such that if the first port is not available but its folding partner is available, a route is set up using the folded port; or when performing a maze routing protocol, such that the folding partners of the PORTRT ports are also considered as potential routing candidates.

10

When a send channel or an input port is requesting a path, the PORTRT bus carries a vector which indicates all the oports from which the routing logic may choose to form the next link in the Xmission path. PORTSRC carries a vector which identifies from which channel the request comes; and the OPALL vector indicates which output ports have already been allocated.

15

For a wormhole routing protocol, a find first one (FFO) is performed with the routing FFO (120) on bits in the PORTRT vector, starting at the first bit position beyond the port (on the PORTSRC bus) from which a request came, and wrapping around to bit position 0 and beyond if necessary (this is a `helical´ search). The first `one´ bit indicates the output port through

20   which the next link in the route must be taken. If folding is enabled for this port (FOLDEN), the folded output port corresponding to this one is also available for routing and is indicated in the output vector of the FFO. This vector is masked with OPORTEN and OPALL to generate the output port select vector on the OPSELV bus. If OPSELV is all zeroes, route selection has failed, i.e. the output port required to route the next link is unavailable, and the send channel or

25   ioport must retry the routing request until the output port becomes available.

For a maze routing protocol, the PORTRT vector is first enhanced with any folded ports as indicated by the FOLDEN register. It is then masked with OPALL and OPORTEN, before the FFO is performed. As with the wormhole case, the FFO operation starts at the first bit position

30   beyond that indicated with the PORTSRC bus and wraps around as needed (`helical´ search). Thus, for maze routing, the first available output port from the PORTRT vector will be selected

-12-

and placed on the OPSELV bus. If none of the PORTRT ports is available, OPSELV will be zero
and the route selections has failed. If the route request was from a Send Channel, the Send
Channel will then place a route-rejection status in its ETE queue and interrupt the CPU. If the
route request was from an input port, a route-rejection command will be sent back to the
5   previous node in the path, via the output port paired with the requesting input port.

Routing Arbiter and Signals

The routing arbiter and signals (20) shown in **FIGURE 1** is a find-first-one (FFO) chain. The last
port or send channel that was selected is saved. The next time a search is made, the search
10   starts just beyond that last position and a round-robin type of priority search is conducted.
Ports or send channels arbitrate for route select to get access to the routing logic (16) which
will select an output port to form the next link in an Xmission path. The cut-through arbiter
and signals logic (18) then is invoked.

15   Cut-through Arbiter and Signals

The cut-through arbiter and signals (18) shown in **FIGURE 1** is a find first one (FFO) chain.
Cut-through port allocation priority is similar to that described in SN 07/587,237, but port
allocation priority is not hard-wired. The last port in the channel that was selected is saved.
The next time a search is made, the search starts just beyond that last position and a round-
20   robin type of priority search is conducted.

Maze Routing

Refer to **FIGURE 7** which is a maze routing example in an eight processor network dimension 3
hyper cube. A source (src) node (000) attempts a transmission to a destination or target node
25   (111) by sending scout packets along route search paths indicated by broken lines. Scout
packets encounter blocked links illustrated by the mark "+". Links from 001 to 101 and from
011 to 111 are blocked in this example. A route ready acknowledge path is illustrated by the
solid line from node 111 through nodes 110 and 010, back to the source node 000. The
message is then sent out to the target node 111 over the path as illustrated by the bold solid
30   line.

-13-

Cut-through hardware

To support multi-path routing, the receive and send channels (12, 14) are logically independent of the communication ports (22). Each send and receive channel therefore needs cut-through logic to direct data to or from the selected port or ports. This cut-through logic is similar to that described in SN 07/587,237, replicated for each of the 18 ports and 8 send and 8 receive channels

Route-rejection logic

As a scout packet searches out a path, it must recognize a blocked or busy channel and respond with a rejection packet that retraces and deallocates each channel in the nodes of the scout packet 's path. The retracing is done using the same baktrak paths and logic used by end-to-end acknowledge (ETE-ack)packets.

Maze route-selection and retry logic

The routes from any node along a path are selected by performing an exclusive OR (XOR) of a destination node-ID with a node ID. This is just like cut-through port selection for a fixed-router as described in SN 07/587,237, but all selected ports not already allocated, are potential paths, rather than just the lowest one. The lowest unallocated port is selected by the routing logic.

A port rejection from an allocated path causes the corresponding cut-through cell to be invalidated and the next selected port to be tried. If no valid port selections remain, a rejection message is passed back to the previous node or Send channel in the path. A scout that arrives successfully at the destination node is directed to the next available Receive Channel and then retraces its path as a path_ack message. If the send channel receives a path_ack message, it can start transmitting the requested message along the selected path. If all potential paths are rejected, the cpu is interrupted, at which time random wait and retry is invoked by the software.

Refer to **FIGURE 8** which is a graph of message latency versus percent of network transmitting data. An adaptive maze router is represented by solid lines and a fixed wormhole router is represented by broken lines. Three message mixes are plotted: small (16 packets) medium (128 packets), and large (1024 packets). The vertical axis is message latency which is the

-14-

number of packet time units to deliver the first packet. The horizontal axis is the percent of the network that is transmitting data, that is, the percent of network bandwidth transmitting data. The message mixes described in **TABLE I** are plotted for both routing types in the graph of **FIGURE 8.**

TABLE I

LINE PLOT	SMALL MESSAGE	MEDIUM MESSAGE	LARGE MESSAGE
A	98%	0%	2%
B	50%	48%	2
C	50%	0	50%
D	0%	0%	100%
E	0&	100%	0%
F	100%	0%	0%

As shown in the graph of **FIGURE 8**, the maze router out-performs a fixed wormhole router in most situations.

Message Protocols

End-to-End reporting

A transmission may optionally hold a path until an end-to-end (ETE) acknowledge is received back at the source node from the destination (target) node. The ETE_ack or ETE_nak is sent back along the same source to target path, but in the reverse direction, from target to source, as the transmission that was delivered. The target to source path uses companion ports that transmit in the other direction along a back-track routing network. The ETE_nak includes error status that indicates "parity_error", "rcv_count_overflow", or "flushed". The ETE_ack includes a 6-bit status field set up by software at the receiver end. ETE packets are not queued behind other messages along the companion path, but are inserted between normal message packets using the back-track routing network. ETE packets are delivered to the send channel, at the source node, that initiated the transmission.

Packet Formats

Messages are delivered via packets of different types. Every data transmission begins with a 32-bit address packet, followed by 72-bit data packets, which include 64 bits of data and 8

-15-

bits of packet overhead. Message data must be double-word aligned in memory. Commands are delivered in 18-bit packets and are generated and interpreted by the hardware only.

Figure 9 is a diagram of an address (scout) packet (32 bits):

5   Start bits   - 2
         Packet type   - 2
         Node Address   - 18
         Forward bit   - 1
         Routing type   - 3
10   Reserved   - 2
         Acknowledge   - 2
         Parity   - 2

Figure 10 is a diagram of a data packet (72 bits):

15   Start bits   - 2
         Packet type   - 2
         Data   - 64
         Acknowledge   - 2
         Parity   - 2

20

FIGURE 11 is a diagram of a command packet (18 bits):

         Start bits   - 2
         Packet type   - 2
         Command   - 4
25   Status   - 6
         Acknowledge   - 2
         Parity   - 2

Routing types:

30   Bit0 indicates "oblivious" routing, i.e. only a single route is possible at
         any intermediate node; non-adaptive.

-16-

Bit1 indicates "progressive" routing, i.e. data wormholes behind the address -

there is no circuit probe(scout packet).

Bit2 indicates "alternate" routing, i.e. mis-route to non-minimum-path

neighbors when further routing is otherwise blocked.

5

000 = "maze" routing: exhaustive back-track using a circuit probe (scout   packet)

001 = "helix" routing: n minimum path oblivious routes tried from the sender, using a circuit

probe

10   010 = RESERVED

011 = oblivious wormhole routing, (the only non-adaptive rt type)

100 = "alternate_maze": maze until source is fully blocked, then mis-route and

maze through neighbor nodes, in turn as necessary

101 = "alternate_helix": helix until source is fully blocked, then mis-route and

15   helix along non-minimum paths

110 = "hydra": maze progressively (take 1st available minimum path port at   each inter-

mediate node, wormholing data behind) until all paths are   blocked, then mis-   route and

maze from blocked node

20   111 = "oblivious_hydra": oblivious wormhole (take only 1st minimum path port   at each

intermediate node, wormholing data behind) until path is blocked,   then mis-route and maze

from blocked node.

25 Packet types:

00 = address

01 = data

10 = bak-trak routing_command(rt_rej,rcv_rej,fwd_rej,rt_ack,ETE_ack,ETE_nak)

11 = fwd_message_command (EOM,EOT,flush,reset)

30

Commands:

-17-

stat cmd
---- ----

    xxxxxx 0000 = packet acknowledge

    xxxxxx 0001 = route ack (path ack)

5   xxxxxx 0010 = route rejected (blocked) (path rejection)

    xxxxxx 0011 = reserved

    xxxxx0 0100 = rcv_channel rejected or hydra_route rejected

    xxxxx1 0100 = parity_err flushed back

    xxxxxx 0101 = forwarded route rejected

10   ssssss 0110 = ETE ack (ssssss = rcv_code)

    ssrrrr 0111 = ETE nack (rrrr= error status; ss=rcv code)

    xxxxxx 1000 = EOM

    xxxxx0 1001 = EOT - no ETE requested

    xxxxx1 1001 = EOT - ETE requested

15   xxxxxx 101x = reserved

    xxxxxx 1100 = reset_CPU

    xxxxxx 1101 = reset_node

    xxxxx0 1110 = flush_path

    xxxxx1 1110 = parity_err flushed forward

20   xxxxxx 1111 = reserved

Node Addressing

Processor IDs and destination addresses are 18-bit unique values, specifying one of 256K possible nodes in the system. The 18-bit physical node address of a target node is included in

25 the address packet at the head of a message transmission, and as part of a "scout" packet (circuit probe) when a maze route is being established.

Logical to physical node address conversion, node address checking, and port_vector calculation for a new Transmission Send are done directly by system software.

30

Message Routing

-18-

A message can be routed any of seven ways, as listed under routing types table in the packet formats section above. A programmer selects the routing method via the routing-type field in the operand of the "Set_Path" or "Send" instruction.

5   Oblivious wormhole routing is a "fixed" routing scheme. A message from a given source to a given destination takes exactly one unique predetermined routing path. The path taken is the lowest-order uphill minimum-length path. The message, node address followed immediately by message data, worms its way toward the destination node, not knowing if the path is free and never backing out, but blocking on busy ports as it encounters them and continuing on when

10   the busy port frees up.

Maze routing in accordance with the present invention is an adaptive routing scheme. For a message from node_A to node_B, all minimum-length paths between the two nodes are searched one at a time (actually, paths in which the first leg is non- minimum may optionally be tried also) by a single-packet scout, starting with the lowest uphill path and doing a depth-

15   first helical traversal of the minimum-path graph until a free path to the destination is found. The successful arrival of a scout packet at the destination establishes the path. Then, once a path_acknowledge packet is delivered back to the sender, this reserved path is used to transmit the message. If no free path is found, however, an interrupt is generated at the source

20   node, whereupon the software may retry the path search after an appropriate delay or use alternate routing (and/or using a different set of first-leg paths).

Maze routing protocol

In a maze router, a transmission from a source (sender) node to a destination (target) node

25   cannot be accomplished until a path is established from a Send Channel at the source node to a Receive Channel at the target node. The path is established as follows:

At the Sender node

Each of the source's Send channels has associated with it a send_port_vector (SPORTVEC),

30   provided to it by the software via a Send instruction, which indicates the output ports of the sender's node through which the routing will initially be attempted. These ports may or may not

-19-

start minimum-length paths. This first hop may thus route non-minimally, while all subsequent hops will take only minimum paths to the target. In other words, the maze router does an exhaustive search of minimum paths between a set of nodes, that set including the source node and/or some number of its immediate accessible neighbors, and the target node.

5

A scout packet, including the node address of the target node, is sent out the first of the source's selected output ports which is enabled and free, and is thus delivered to a neighboring node, the next node in a potential path to the target node. The path from the Send Channel to the selected output port is now locked, reserved for the pending transmission,

10   unless a path_rejection packet is subsequently received on the corresponding input port. If the selected output port receives a path_rejection packet, because all paths beyond the next node are blocked, a new output port from the send_ port_vector will be selected, if available, and the scout packet sent out that port. When no more send_port_vector output ports are available, because they were blocked or rejected, an "all_paths_blocked" status is pushed into the ETE

15   queue for the respective Send channel, the CPU is interrupted, and the Send channel goes into a wait state, waiting for software to clear it. If, however, a path_acknowledge packet is received, it is passed back to the Send_DMA_Channel that initiated the search and the selected path remains reserved for the subsequent transmission, which can now be started.

20   At the Target node

A node that receives a scout packet at one of its input ports first compares the target node address from the scout packet with its own node ID. If they match, the scout packet has found the target node. If a receive channel is free, the scout packet is delivered to it and a path_acknowledge packet is sent all the way back to the source (sender) node, retracing the

25   successful legs of the scout's path. If a receive channel is not available, a path_rejection packet, encoded as a "rcv_ channel_ unavailable" command, is sent back to the source node via the established path and the input port is freed-up.

At Intermediate nodes

30   If a receiving node's node ID does not match the target node address, then this node is an intermediate node and it will attempt to deliver the scout packet to the next (neighboring) node

-20-

along a minimum-length path to the target node. The XOR of the target node address with this node's node ID, the Hamming distance between them, indicates which output ports connect to minimum paths, and is latched in the IPORTVEC register as the cut-through vector. The output port paired with this input port, i.e. the link back to the node from where the scout just came,

5   is disqualified from the cut-through vector, thus preventing any cycles that could be caused by non-minimal routes (which are allowed on the first hop). If folding is enabled, the bits corresponding to the folded partners of the cut-through vector are also asserted. The scout packet is sent out the first of the cut-through output ports, starting beyond this input port, which is enabled and free. The path from the input port to the selected output port is reserved

10   for the pending transmission, unless and until a path_rejection packet is received on the output port's companion input port. If a path_rejection packet is received, because all minimum paths beyond the next node are blocked, a new cut-through port will be selected, if available, and the scout packet sent out that port. When no more cut-through ports are available from this node, a path_rejection packet is sent to the previous node, the one from which the scout packet got

15   here, and the input port is freed-up. If, however, a path_ acknowledge packet is received, it is passed back to the source node via the established path and the selected path remains reserved for the subsequent transmission.

The above process is continued recursively until a free path to the target is found and

20   established, or until all desired paths from the source node have been tried and failed.

Path_Cmd packets:

A scout returns path_rejection status to the previous node, or path_found status to the source node, by sending back a path_cmd packet. Path_cmd packets are sent back along a path

25   using the path's "companion" ports, just like an ETE packet. There are two kinds of path_cmd packets. A "path_acknowledge" packet, indicating that the scout has established a path to the destination node, is delivered all the way back to the source, leaving the path established for the subsequent transmission. A "path_rejection" packet, indicating that the scout has been completely blocked at an intermediate node, is delivered to the previous node in the path,

30   clearing the path (this last hop) along the way. A new path from that node may now be tried or, if no new paths remain open from that node, it will in turn send a "path-rejection" packet to

-21-

its antecedent node. If it has no antecedent node, i.e. it is the source node, the rejection packet is placed into the ETE queue, the Send DMA channel goes into a wait state, and the CPU is interrupted.

5   Routing Retry using Alternate Send_Port_Vector

If the routing logic fails to find a path using the given send_port_vector, an alternative set of paths may optionally be attempted before interrupting the CPU.

When alternate routing is enabled, and after the initial set of routes has failed, the initial
10   send_port_vector is inverted and and'ed with the alternate_port_mask to create a new send_port_vector. Then, a second attempt is made at finding a route, through neighboring nodes that were not used in the initial try. If the alternate routes also fail, the CPU is then interrupted in the usual manner.

15   Non-minimum paths through alternate send ports are exactly two hops longer than minimum, since all routing is minimum after the first hop. If a source and destination node are separated in j dimensions, the minimum path distance is j hops and the alternate path distance is $j+2$ hops.

20   Attempting alternate routes can be especially important for transmissions to target nodes that are only a short distance away. For example, there is only one minimum-length path to a connected neighbor, yet by attempting routes through all the other neighbors, there are a total of n unique paths to any nearest neighbor in a cube of dimension n as described by the alternate mask.
25
There is one Alternate_Port_Mask per node, but alternate routing is enabled on a per-transmission basis (a bit in the path-setup operand of the SEND instruction).

Folding
30   Folding increases the number of output ports available for routing a message in a non-maximum-size system. Any of the connections from the lower 8 output ports to the

-22-

corresponding input ports of nearest_neighbor nodes, can be duplicated on the upper 8 output ports, in reverse order, to the same nearest_neighbor nodes. In other words, any subset of the interconnect network can be duplicated on otherwise unused upper ports.

5   If folding is enabled (see FOLDEN_register, figure 6), then when a port vector (PORTVEC) is calculated at an intermediate node, any selected ports that are folded will enable their respective companion ports to also be selected into the port vector.

At any hop of a wormhole route, either of the two folded ports, that duplicate the link for the
10   desired dimension, may be used. Folding thus greatly improves the chances of a wormhole route finding its way to the target with minimal or no blocking.

For a maze route, folding increases the number of minimum-path links that can be tried at each hop, and thus improves the chances of finding an open path.
15

Forwarding
The maze router finds a route to the forwarding node, reserves that path, then transmits the next address (fetched from the message data) to that node, whereupon the address is maze-routed from there to the new node. This can be repeated as long as new addresses are
20   forwarded, or until a route cannot be found, in which case the entire path is unraveled and deallocated and a "forward_route_rejected" command is delivered to the send channel's ETE queue. On the other hand, if a path to the final target node is established, the message data is then transmitted normally from the source to the target.

25   Communication Direct Memory Access (DMA) Channels
A message is transmitted from a contiguous block of physical memory at the sender to a contiguous block of physical memory at the receiver, in increments of double-words (64 bits). To provide memory access and message and path control at both ends of the transmission, there are eight Send DMA Channels and eight Receive DMA Channels at each processor.
30

-23-

DMA channels are set up with the appropriate SEND or RECEIVE instruction. A Set_DMA
instruction is also provided to assist in setting up the DMA operand of the SEND or RECEIVE
instruction. The SEND and RECEIVE operands provide path control, messaging parameters,
addresses, etc. for the DMA channels and routing logic.

5

In order to reduce page-mode page-break limitations on DMA memory bandwidth, each
channel, send or receive, buffers up to 32 bytes of data. This corresponds to 4 double-word
(64-bit) memory accesses. Messages must be aligned on double-word boundaries and sized in
double-word-multiples.

10

Send_DMA
Each Send channel has associated with it a physical memory address and a message length,
stored in its DMA register, as well as a destination node ID and a send_port_vector, stored in
its Path register. The Send channels are double-buffered, such that the DMA and Path control
15   descriptors of the next message can be setup while the current one is being transmitted.
Communications software can use this feature to hide messaging overhead and to efficiently
implement send-chaining.

After a Send channel has been setup for a new transmission, it first enters the routing state to
20   establish a path to the target node. The path is established once the address packet is
transmitted to the output port, if routing progressively, or when a path_acknowledge packet is
received by the channel, if routing maze.

If the node address is forwarded, the send channel enters the forwarding state and transmits
25   address packets from the message data until the last address packet is not marked as
forwarded. If routing maze, the channel waits for a path_acknowledge after each address is
transmitted.

Once a Send channel establishes a path to the target node , it commences reading the
30   message data from memory and transmitting it along the path to the target node. As the
message data is fetched, the memory address is incremented and the message length is

-24-

decremented, until the length counter reaches zero. When the send counter reaches zero, an End-of-Message (EOM) or End-of-Transmission (EOT) packet is sent, depending on the EOT-enable bit of the channel setup.

5   If it's an EOM, the DMA register is cleared and a new one popped in from the Send buffer. If it's an EOT and ETE is not enabled, the DMA and Path registers are both cleared and reloaded from the Send buffer. If it's an EOT and ETE is enabled, the Send channel is not cleared in any way, but waits for the ETE packet. When the ETE packet arrives, it is pushed into the ETE Queue, and the Send channel (both registers) is cleared. The Send channel then moves on
10   directly to the next transmission (pops the Send buffer) if it's ready. Whenever the Send buffer is popped due to an EOM or EOT condition, the CPU is also interrupted to indicate that a free Send channel is now available. ETE also generates an interrupt if interrupt is enabled.

When maze routing, the ETE queue is also pushed with status information if a route could not
15   be found to the target node. In this case, the path_rdy bit is cleared, an ETE interrupt is raised, but the DMA channel is not popped, cleared, or reloaded. A programmer can subsequently clear the Send channel by writing to the corresponding DMA register.

An ongoing Send transmission can be stopped by clearing the DMA_rdy bit in the channel's
20   DMA register. This stops the transmission, but leaves it in the transmitting state. The DMA_rdy bit can be cleared by writing a 1 to the respective bit, corresponding to the send channel, of the Send_rdy register (see Send Channel Status Registers).

A blocked or stopped Send transmission can be flushed by writing a 1 to the respective bit,
25   corresponding to the send channel, of the Send_transmission_rdy register (see Send Channel Status Registers).

When a message is flushed, a flush-cmd packet traverses the allocated path, clearing and deallocating the path behind it.
30

-25-

End-to-End Queue

For each Send channel there is an End-to-End (ETE) Queue, into which ETE status, from the target node's receive channel, or route_rejection or error status is pushed. When status is pushed into the ETE queue, an ETE interrupt is generated. The queue is 2 entries deep and a
5   processor register, one for each send channel, contains both entries. A programmer can read an ETE queue, without side effects, via a RDPR instruction. The programmer can then clear an ETE entry by writing a zero into its valid bit, via a WRPR instruction (though they must be read together, each entry in the queue can be written separately). When the first entry is cleared (popped) in this way, the second entry is automatically copied into its place and cleared. The
10 Send channel cannot start a new transmission while the ETE Queue is full.

Send Operation

FIGURE 12 is a flow diagram of a send operation. From an idle state, the send channel enters the routing state (200). The first unallocated output port is selected from the send port vector
15 (202). If a port is selected (204), the flow proceeds to block (206). The send channel allocates the selected port, and sends the address packet out of the selected output port (208). The send channel then waits for routing status to be returned via the route command (210).

20 When the route command arrives, the status (212) is either route rejected or route established.

If at block (212) the status is route rejected, the send channel clears the corresponding bit in the send port vector, clears port select, and deallocates the output port it had allocated at block (206). If the send port vector is now reduced to 0, and alternate routing is not enabled
25 (205), or if enabled but this is not the first pass (207) through the sequence, the send channel pushes route_rej status onto the ETE queue and if interrupt is enabled, the send channel interrupts the CPU (218). The send channel then enters the idle state (220).

If at block (212) the route is established, route_ready is set (222) and the forward bit is
30 checked (223). If the forward bit is set, the forwarding state is entered (225). If not, the enter

-26-

message transmission state is entered (224). The send channel transmits data to the target node until the message count is 0.

5   If at block (204) a port is not selected, the flow proceeds to.). decision block (205). If alternate routing is enabled, and this is a first pass through the flow sequence (207), the SPORTVEC is made equal an inverted version of the initial send_port vector (209). Thus when all initially attempted routes fail using the initial SPORTVEC, the inverted version provides an alternate route select attempt as the flow proceeds to block (202). The first unallocated output port is selected from the now inverted send port vector (202). If a port is selected (204), the
10 flow proceeds to block (206). If a port is not selected (204), the flow proceeds to block (205). Alternate routing is enabled (205), but this is not the first pass (207) through the sequence, so the flow proceeds to block (218).The send channel pushes route_rej status onto the ETE queue and if interrupt is enabled, the send channel interrupts the CPU (218). The send channel then enters the idle state (220).

15

Receive_DMA
Each Receive channel has associated with it a physical memory address and a message length (also called the receive count), stored in its respective DMA register. It also has a rcv_status register that includes error status and the receive code. As a message flows through the
20 channel, the address increments and the message length decrements, until the length counter reaches zero or until an EOM/EOT packet is received.

If a Receive channel receives an EOM or EOT before the counter has reached zero, or immediately after it reached zero, the message has successfully completed and the channel
25 returns to the idle state, clearing dma_rdy. If no receive errors occurred during the reception, a rcv_rdy interrupt is raised. Otherwise, a rcv_err interrupt is raised.

For example, if a parity error is detected anywhere along the transmission path, a parity_err flush_message is delivered forward to the receive channel of the target (as well as back to the
30 send channel of the sender). The parity error or flush bits in the receive status field are set and the target CPU is interrupted with a rcv_err interrupt by the receive channel.

-27-

If the receive counter reaches zero, the message should be complete and the next packet should be an EOM or EOT. If it is not, the rcv_count_overflow flag in the receive status field is set, and all further packets are ignored, i.e. simply shifted into oblivion, until an EOM or EOT is received, at which point a rcv_err interrupt is generated. The counter wraps and continues to
5   decrement (the address does not increment), thus providing a way for a programmer to calculate how far the message overflowed.

A programmer can read the receive status, message count, etc. at any time, by simply reading the processor registers associated with the channel.
10

Scatter/Gather at the Receive Channel

To facilitate fast "gather" functions at the receiver, the programmer can optionally set the "ignore_EOM" flag at the receive channel for a given transmission (see Receive instruction description). Thus, the sender may gather disjoint bundles of data, as individual messages,
15   into a single transmission, and the receiver can be set up to ignore the message boundaries for the length of the entire transmission, and thus store the bundles sequentially in a single DMA operation, rather than taking an interrupt and setting up a new receive_DMA after every message.

20   To implement a "scatter" function, the programmer can optionally set the "force_EOM" flag at the receive channel. Thus, the sender may deliver a sequential block of data in one message, and the receiver can be set up to force message boundaries for sub-lengths of the transmission, and thus scatter the data in sub-blocks to different areas in memory. The receive channel is set up with a length shorter than the incoming message, and when the length
25   counter drops to zero, the receive channel treats it as an EOM and blocks the incoming data until new DMA parameters are set up by the programmer. This is especially useful for DMAihg a message across virtual page boundaries that may map to disjoint physical memory pages.

30   Routing From an Input Port

-28-

FIGURE 13 is a flow diagram an address packet input port operation. The input port receives an address packet (300) and computes the exclusive OR of the address in the address packet with the Node ID of this node (302). The result is ID_diff. If ID_diff is 0 or if the input port is designated as a terminal, then the flow proceeds to block (322). If not, then the flow proceeds
5   to block (306).

At block (306) the port vector (portVec) is generated and used to select the first unallocated output port (308).

10   At block (310), if a port is not selected, then the input port sends a route_reject command via the output port paired with this input port (335), and waits for a new address packet (336).

If a port is selected (310), then an address packet is forwarded to the next node via the selected output port (312) and the port is allocated. The transmission path through this node is
15   now setup and the input port waits for routing status that will be supplied by an incoming route command (314). A route command (316) will either indicate that the route is rejected or that the route is ready. If rejected, the flow proceeds to block (318). If ready, the flow proceeds to block (330).

20   At block (318), the receive channel clears the corresponding bit in the port vector, clears port select, and deallocates the output port allocated at block (312). The input port selects the next unallocated output port from the port vector (308) via the routing logic,and the flow proceeds as described above.

25   At decision block (304), if the node ID is equal to the address in the address packet or this port is terminal, then this node is the target node and the flow proceeds to block (322).

At block (322) the port vector (portVec) is generated and used to select the first ready receive channel (324). If a channel is selected (326), then the input port allocates a receive channel to
30   receive the message (328). The input port sends a route ready (route_rdy) command via the output port paired with this input port (330) and waits for message data to arrive (332).

-29-

At block (326), if a channel is not selected, then the input port sends a route_reject command via the output port paired with this input port (335) and waits for a new address packet (336).

End to End Reporting

5   FIGURE 14 illustrates end-to-end acknowledge. At the source node (350), the send channel sends a message packet out of an output port (352) to an intermediate node (355) that receives the message at an input port (354). The message is sent by the intermediate node (354) out of an output port (356). The message travels from node to node until the target node (358) receives the message packet. A receive channel is allocated (362) and an ETE ack

10   message is sent back over the same path by using the output ports that are paired with the respective input ports in the path (ports 361, 353, and 351). The message path is held until the ETE ack is received at the source node and receive status is returned with the ETE ack. For each Send channel there is an End-to-End (ETE) Queue, into which ETE status is pushed. When End-to-End status is pushed into the ETE queue, a Send_rdy and ETE interrupt are generated,

15   depending on the status.

FIGURE 15 is a maze route timing diagram wherein there are no blocked links.

FIGURE 16 is a maze route timing diagram wherein there are blocked links and wherein

20   backtracking is invoked.

While the invention has been particularly shown and described with reference to preferred embodiments thereof, it will be understood by those skilled in the art that the foregoing and other changes in form and detail may be made therein without departing from the spirit and

25   scope of the invention.

-30-

What is claimed is:

1. In a network of interconnected nodes;

each node including a processor;

5   each of said processors in said network being assigned a unique processor

identification (ID);

an apparatus for establishing a communication path through a node of said network

comprising:

a plurality of input ports;

10   a plurality of output ports;

control means at one of said input ports of a said node for receiving address packets

related to a current message transmitted from an output port of another of said nodes;

a router connected to said one input port and to said output ports;

registering means for registering said processor identification (ID);

15   comparing means connected to said control means and to said registering means for

comparing a target node address in an packet with said processor ID of said node stored in

said registering means;

said comparing means including means for creating a first condition provided that said

ID is equal to said target node address and, alternatively, a second condition provided that said

20   ID is not equal to said target node address;

a plurality of receive channels connected to said router;

allocation means connected to said comparing means and to said receive channels for

allocating to said one input port, said receive channel of said node upon occurrence of said

first condition that said node address is equal to said target node address; and,

25   first means connected to said comparing means and to said control means for sending

a route_ready command over said output port paired with said input port upon occurrence of

said first condition.

2. The combination in accordance with claim 1 further comprising:

30   a port vector;

second means connected to said comparing means and to said router for selecting a first output port from said port vector upon occurrence of said second condition that said node address is not equal to said target node address; and,

third means connected to said second means and to said input port for forwarding said

5   address packet to a next node over said selected first unallocated output port.

3. The combination in accordance with claim 1 further comprising:

means for selecting a first unallocated output port connected to a third unallocated node of said second node; and,

10   means for forwarding said address packet to said third unallocated node via said selected first output port.

4. The combination in accordance with claim 2 further comprising:

means for selecting a first unallocated output port connected to a third node; and,

15   means for forwarding said address packet to said third node via said selected first output port.

5. In a network of interconnected nodes;

each node including a processor, and a plurality of input ports and a plurality of output

20   ports, each one of said input ports being paired with a corresponding one of said output ports;

each of said processors in said network being assigned a unique processor identification (ID);

a method of establishing a communication path through a node of said network

25   comprising steps of:

A. receiving at an input port of a first of said nodes a first address packet having a target node address therein, said first address packet being related to a current message sent from an output port of a second of said nodes;

B. comparing at said first node said target node address in said first address packet

30   with a processor ID of said first node;

-32-

C. allocating said receive channel of said first node to receive a message upon a first condition that said processor ID is the same as said target node address; and,

D. sending a route_rdy command to said second node over said output port paired with said input port at said first node.

5

6. The method in accordance with claim 5 comprising the further steps of:

E. selecting a first output port connected to a third node of said first node; and,

F. forwarding said address packet to said third node via said selected first output port.

10   7. The method in accordance with claim 6 comprising the further steps of:

G. receiving a route command;

H. sending a route_reject command via said output port paired with said input port.

8. A communication apparatus comprising:

15   a maze router mechanism;

a fixed-router mechanism;

mode selection means connected to said maze router mechanism and to said fixed-router mechanism for selecting either said maze router mechanism or said fixed-router mechanism; and,

20   an input port connected to said maze router mechanism and to said fixed-router mechanism for receiving a node address packet that includes control information that enables the finding and establishing of a route to a destination node from a source node.

9. A communication apparatus comprising:

25   a memory address bus;

a memory data bus;

a direct memory access buffers and logic block is connected to said memory address bus and to said and memory data bus;

a plurality of receive channels;

30   a plurality of send channels;

-33-

a plurality of routing registers connected to said send channels and to said receive channels;

a cut-through arbiter connected to said plurality of receive channels and to said plurality of send channels;

5   a routing arbiter connected to said plurality of send channels and to said routing registers;

a processor bus connected from a central processing unit to said receive channels, said send channels and said routing registers and logic;

a plurality of input ports;

10   a plurality of input pins;

said plurality of input ports being connected to said routing arbiter and to said input pins;

a plurality of output ports; and,

a plurality of output pins;

15   said plurality output ports being connected to said cut-through arbiter and to said output pins;

said plurality of input ports and said plurality of output ports being connected to said cut-through arbiter, said routing arbiter, and to said send and receive channels.

20   10. The communication apparatus in accordance with claim 9 further comprising:

mode selection means connected to said cut-through arbiter and to said routing arbiter for selecting either said cut-through arbiter or said routing arbiter;

one of said input ports including means for receiving a node address packet that includes control information that enables the finding and establishing of a route to a destination

25   node from a source node.

11. A method comprising steps of:

A. selecting a first unallocated output port from a send port vector to provide a selected output port;

30   B. allocating said selected output port to a send channel;

C. sending an address packet out of said selected output port to a target port;

-34-

D. receiving at said send channel a route command containing routing status returned from said target port, said routing status specifying either route rejected or route established;

E. clearing a bit corresponding to said selected output port in said send port vector upon a condition that said routing status is route rejected; and,

5   F. deallocating said selected output port from said send channel.

12. The method in accordance with claim 11 comprising the further steps of:

G. pushing route_rej status onto an end-to-end (ETE) queue upon a condition that said step E results in said send port vector being reduced to zero.

10

13. The method in accordance with claim 11 wherein said address packet includes a forward bit, said method comprising the further steps of:

G. setting a route_ready status upon a condition that a route is established;

H. entering a forwarding state upon a condition that said forward bit is set in said

15   address packet; and,

I. entering a message transmission state upon a condition that said forward bit is not set in said address packet.

14. A method comprising steps of:

20   A. receiving an address packet at an input port, said input port being part of a node having a node ID, said address packet including an address;

B. computing an exclusive OR of said address in said address packet with said Node ID, a result of said exclusive OR being a port vector;

C. selecting a first unallocated output port specified in said port vector to provide a

25   selected output port upon a condition that said result is not equal to zero;

D. forwarding an address packet to a next node via said selected output port;

E. receiving routing status at said receive channel via an input port, said routing status being supplied by an incoming route command, said route command indicating either that a route is rejected or that said route is ready;

30   F. clearing a bit in said port vector corresponding to said output port upon a condition that said route command indicates that a route is rejected; and,

-35-

G. deallocating said output port from said input port.

15. The method in accordance with claim 14 comprising the further steps of:

I. sending a route_rdy command via an output port paired with said input port upon a
5   condition that said route command indicates that a route is ready.

16. The method in accordance with claim 14 comprising the further steps of:

I. sending a route_reject command via an output port paired with said input port upon
a condition that said step C of selecting a first unallocated output port specified in said port
10   vector to provide selected output port results in no port being selected.

17. The method in accordance with claim 14 comprising the further steps of:

I. generating a port vector upon a condition that said result is equal to zero;

J. selecting a first ready receive channel using said port vector to provide a port select
15   vector;

K. selecting a first unallocated input port specified in said port select vector to provide
selected output port;

L. allocating said first ready receive channel to receive a message upon a condition
that an input port is selected; and
20   M. sending a route_rdy command via an output port paired with said input port.

18. The method in accordance with claim 11 comprising the further steps of:

G. inverting said send port vector to provide an alternate send port vector upon a
condition that said routing status is route rejected; and,
25   H. using said alternate send port vector to select a first unallocated output port from
said alternate send port vector to provide said selected output port.

19. A method of transmitting a message through a node comprising steps of:

A. storing a fold enable vector in a fold enable register, said fold enable vector
30   indicating by identification bits which ports are selected to be folded ports;

B. receiving a message packet that includes address bits at an input port of said node;

-36-

C. shifting said address bits into an address field of an input port command register;

D. comparing said address field with an identification address of said node resulting in a difference vector;

E. loading said difference vector into an identification difference register;

5   F. asserting a FOLDEN line upon a condition that folding is enabled;

G. modifying said difference vector with said identification bits corresponding to said folded ports;

H. loading said contents of said identification difference register into an input port vector register;

10   I. using contents of said input port vector register to identify a next minimum path port through which a message may be routed;

J. calculating a cut-through vector at an intermediate node; and,

K. storing a port cut vector indicating which port, send channel or receive channel to cut through.

15

20. A method of message transmission between a plurality of nodes comprising steps of:

A. pairing each of a number of input ports at each node with an associated output port of each node;

B. allocating an originating node send channel at an originating node;

20   C. sending a message packet out of an originating node output port selected by said originating node send channel to a first intermediate node input port that is connected to said originating node output port, said originating node output port being paired with an originating node input port;

D. receiving said message at said first intermediate node input port, said first 25   intermediate node input port being paired with a first intermediate node output port;

E. connecting a second intermediate node output port to said first intermediate node input port, said second intermediate node output port being paired with a second intermediate node input port;

F. connecting said first intermediate node input port to said second intermediate node 30   output port;

-37-

G. receiving said message at a target node input port connected to said second intermediate node output port, said target node input port being paired with a target node output port that is connected to said second intermediate node input port;

H. allocating a target node receive channel at said target node;

5   I. composing, at said target node receive channel, an end_to_end acknowledge message containing receive status;

J. sending said end_to_end acknowledge said target node output port paired with said target node input port, upon a condition that a message transmission has completed.

10   21. The method in accordance with claim 20 comprising further steps of:

K. disconnecting said second intermediate node output port from said first intermediate node input port upon receipt of said end_to_end acknowledge message.

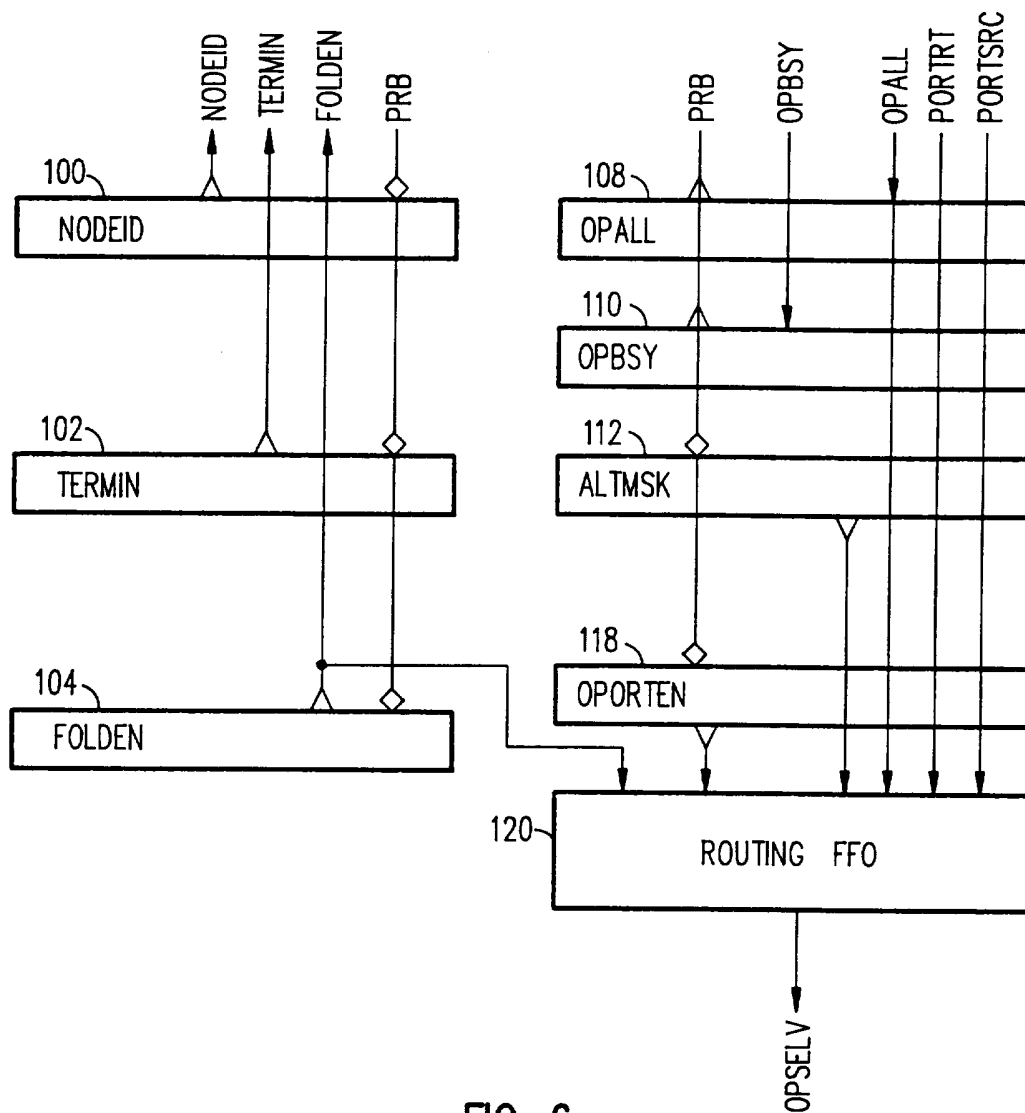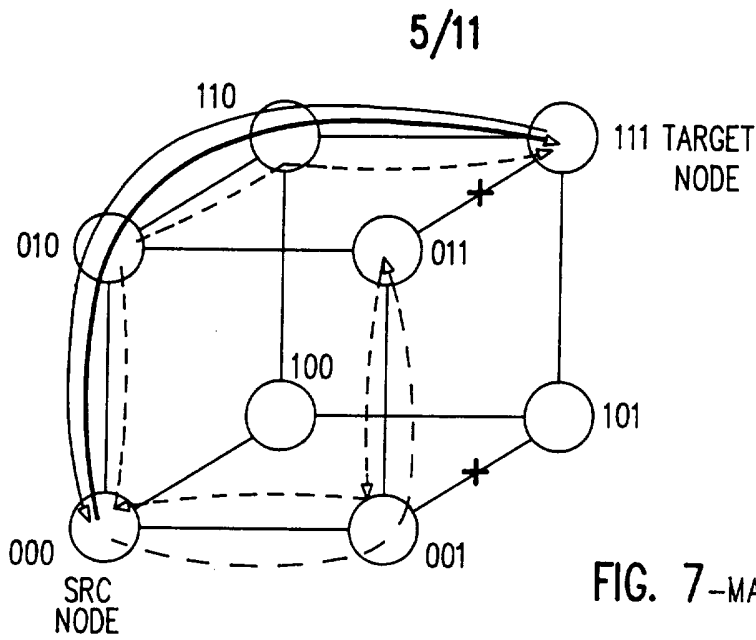1/11
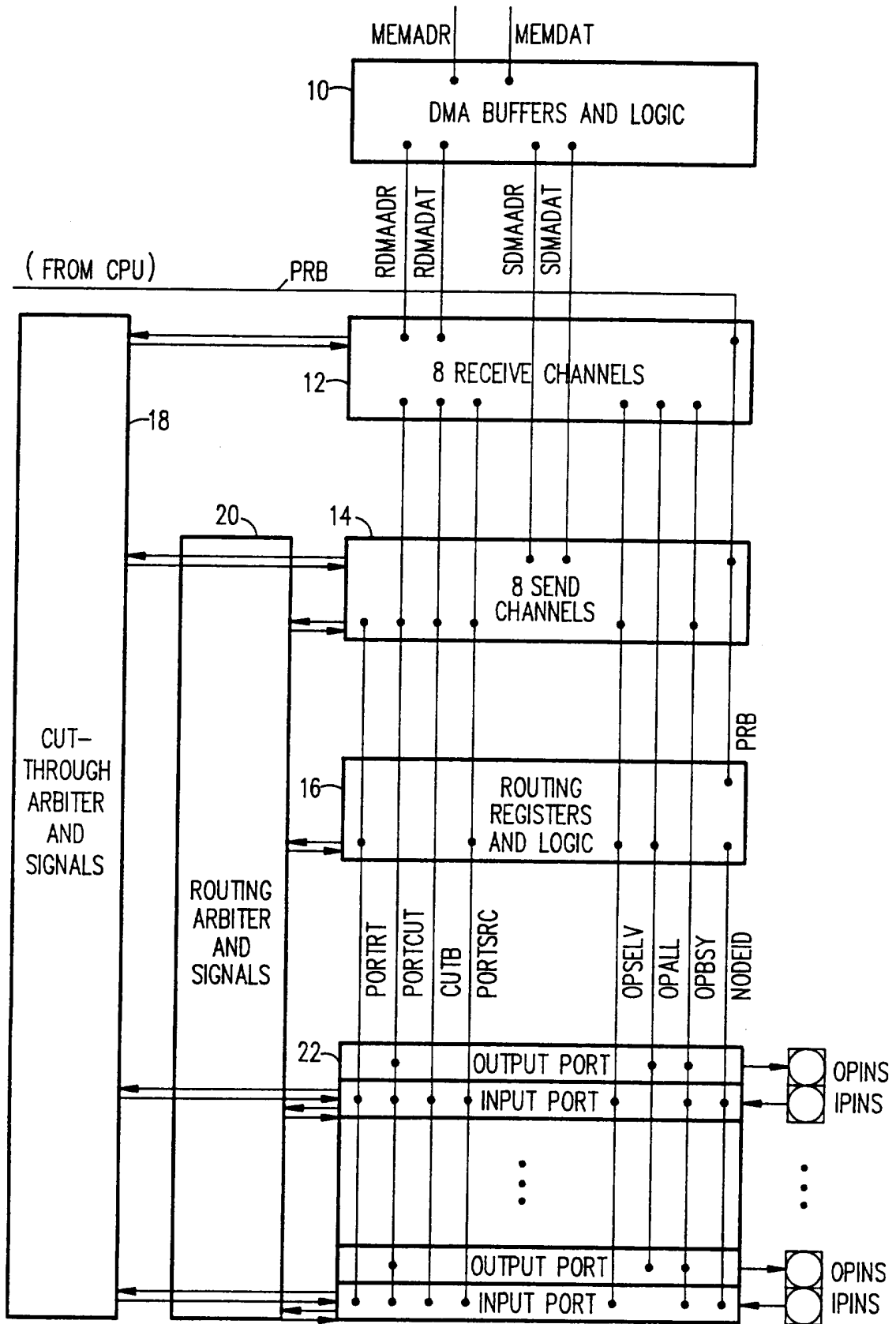
FIG. 1

2/11
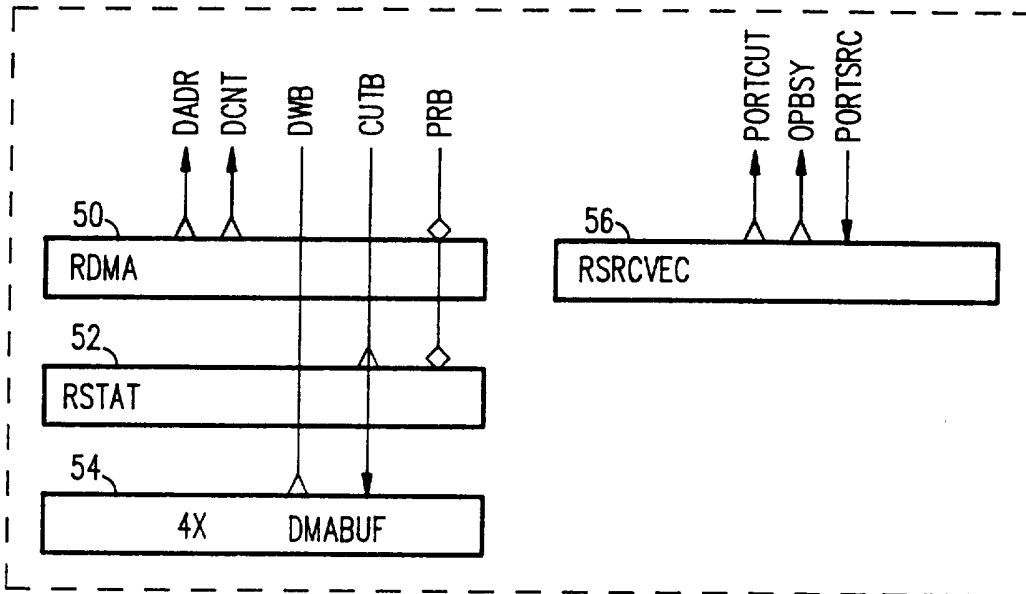
FIG. 2
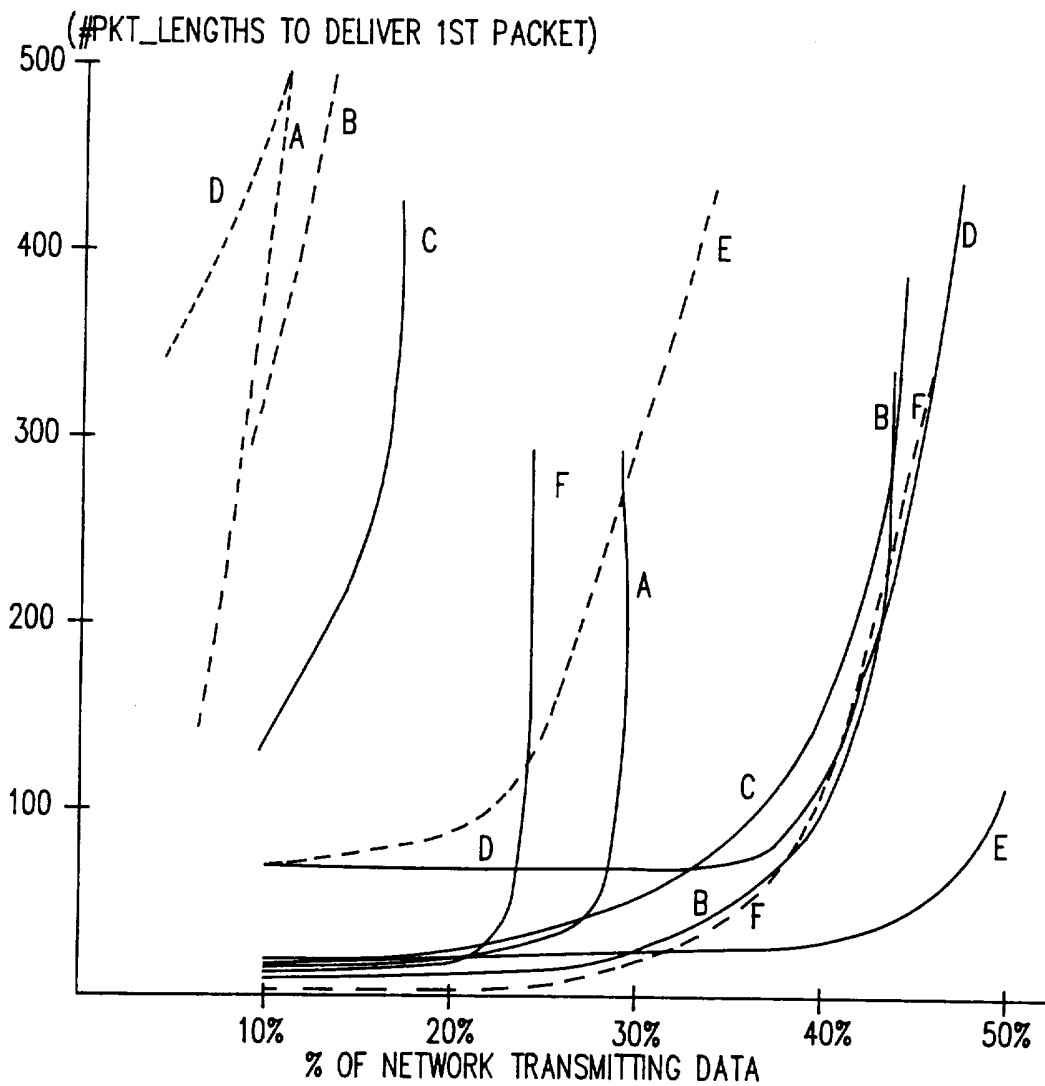

FIG. 3

3/11

FIG. 4

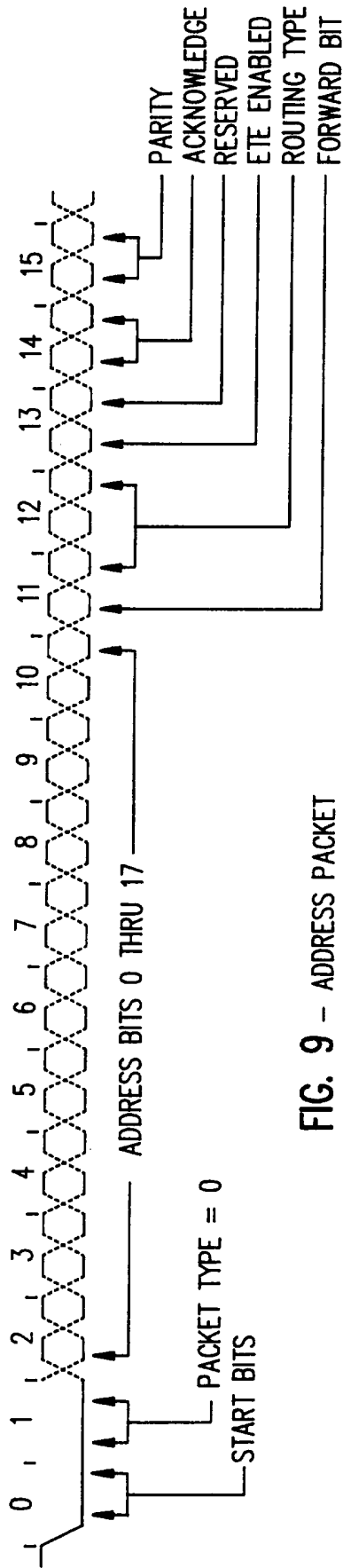

FIG. 5

FIG. 6

5/11
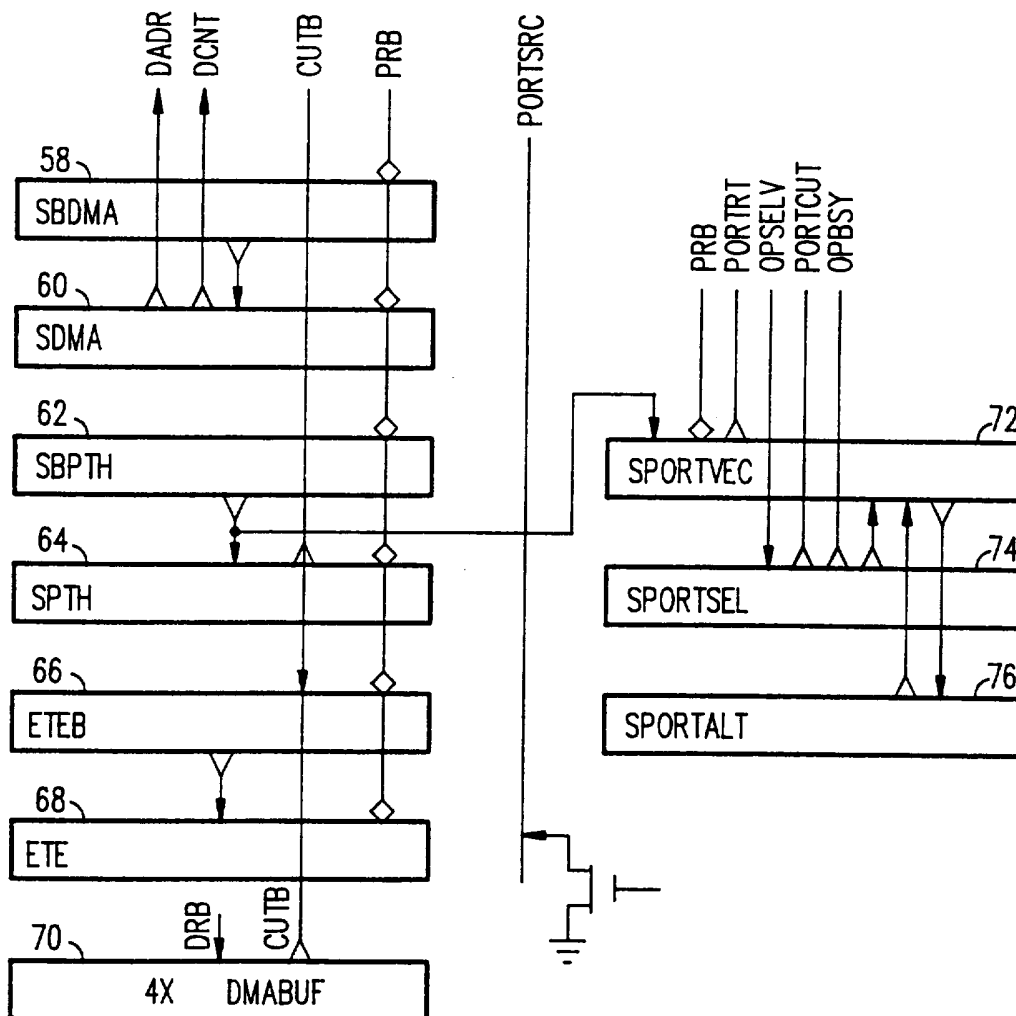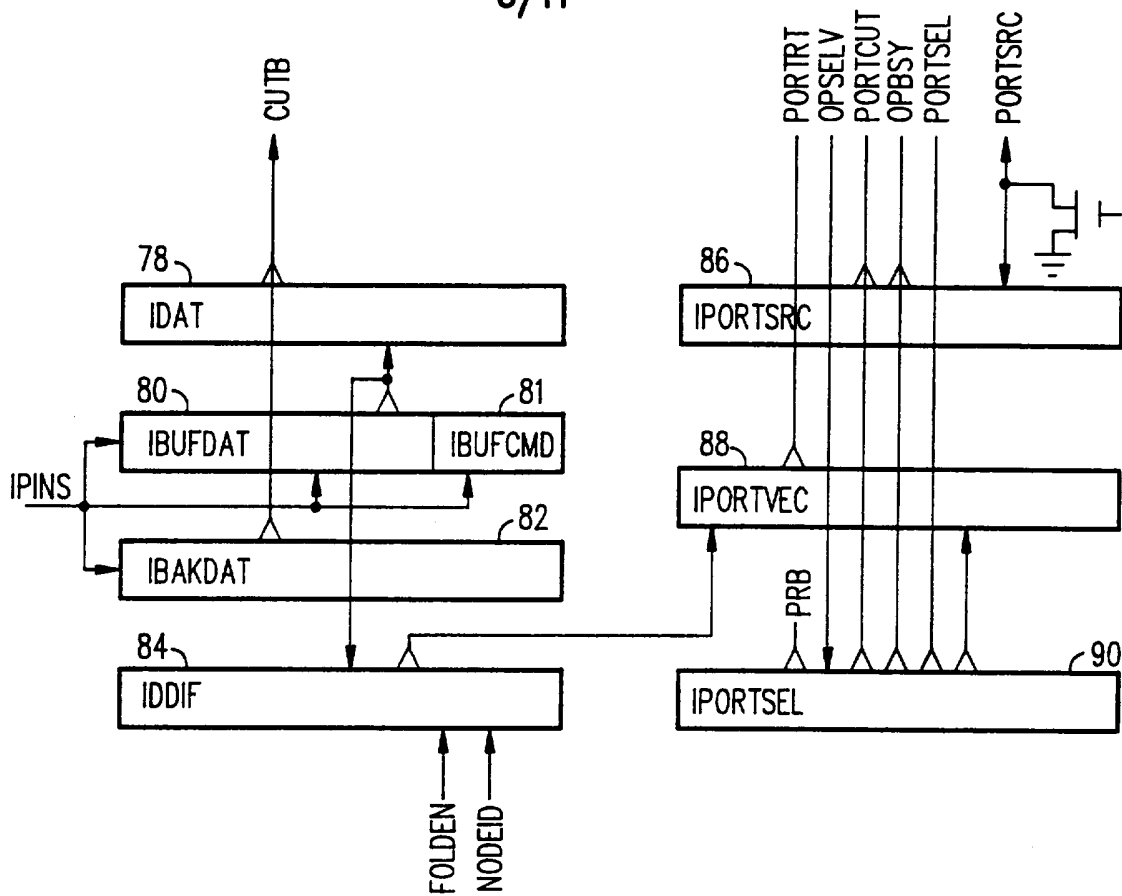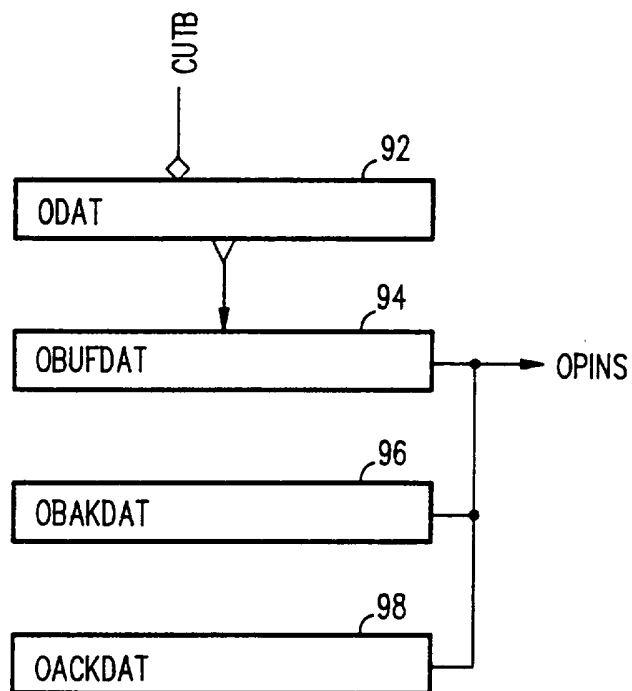
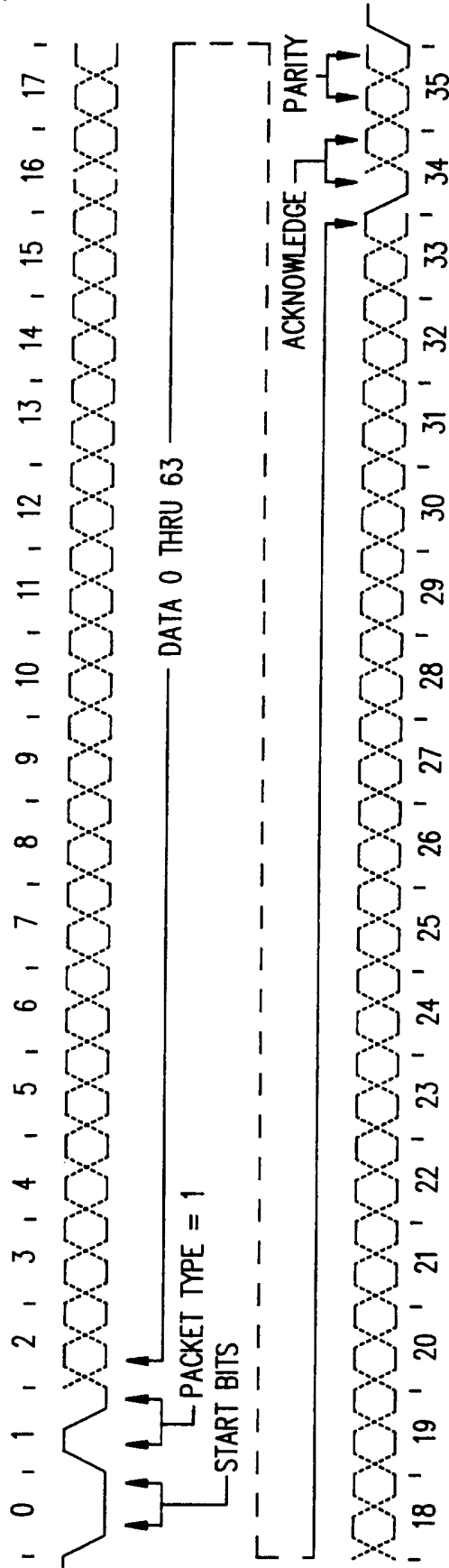FIG. 7-MAZE ROUTING EXAMPLE

FIG. 8

FIG. 9 – ADDRESS PACKET

FIG. 10 – DATA PACKET

FIG. 11 – COMMAND PACKET

8/11

FIG. 12

9/11

FIG. 13

INPUT PORT RECEIVES
ADDRESS PACKET — 300

↓

ID_DIFF =
ADDRESS XOR MY_NODEID — 302

↓

ID_DIFF
=0 OR
THIS_PORT_IS
TERMINAL
? — 304

YES (THIS IS THE TARGET NODE) →

NO ↓

PORTVEC = ID_DIFF — 306

322 — PORTVEC = BITS
CORRESP.
TO RCV_DMA
CHANNELS

↓

308 — PORTSEL = FIRST UNALLOCATED
OUTPUT PORT FROM PORTVEC
(MINUS BIT CORRESPONDING TO
THIS PORT)

324 — PORTSEL = FIRST
READY RCV
CHANNEL

↓

PORT
SELECTED
? — 310

NO → ← NO

CHANNEL
SELECTED
? — 326

YES ↓

328

YES ↓

312 — FORWARD ADDRESS PACKET
TO NEXT NODE VIA
SELECTED OUTPUT PORT;
ALLOCATE PORT

ALLOCATE RCV
CHANNEL TO
RECEIVE MESSAGE

↓

314 — WAIT FOR ROUTING
STATUS VIA
ROUTE_CMD

↓

318 — CLEAR CORRESP. BIT
IN PORTVEC;
CLEAR PORTSEL;
DEALLOCATE OUTPUT PORT

← ROUTE
REJECTED

ROUTE
CMD
? — 316

ROUTE
READY →

330

SEND ROUTE_RDY CMD
VIA OUTPUT PORT
PAIRED W/ THIS
INPUT PORT

↓

332 — WAIT FOR
MESSAGE DATA

335 — SEND ROUTE_REJECT
CMD VIA OUTPUT PORT
PAIRED W/ THIS INPUT PORT

→

336 — WAIT FOR NEW
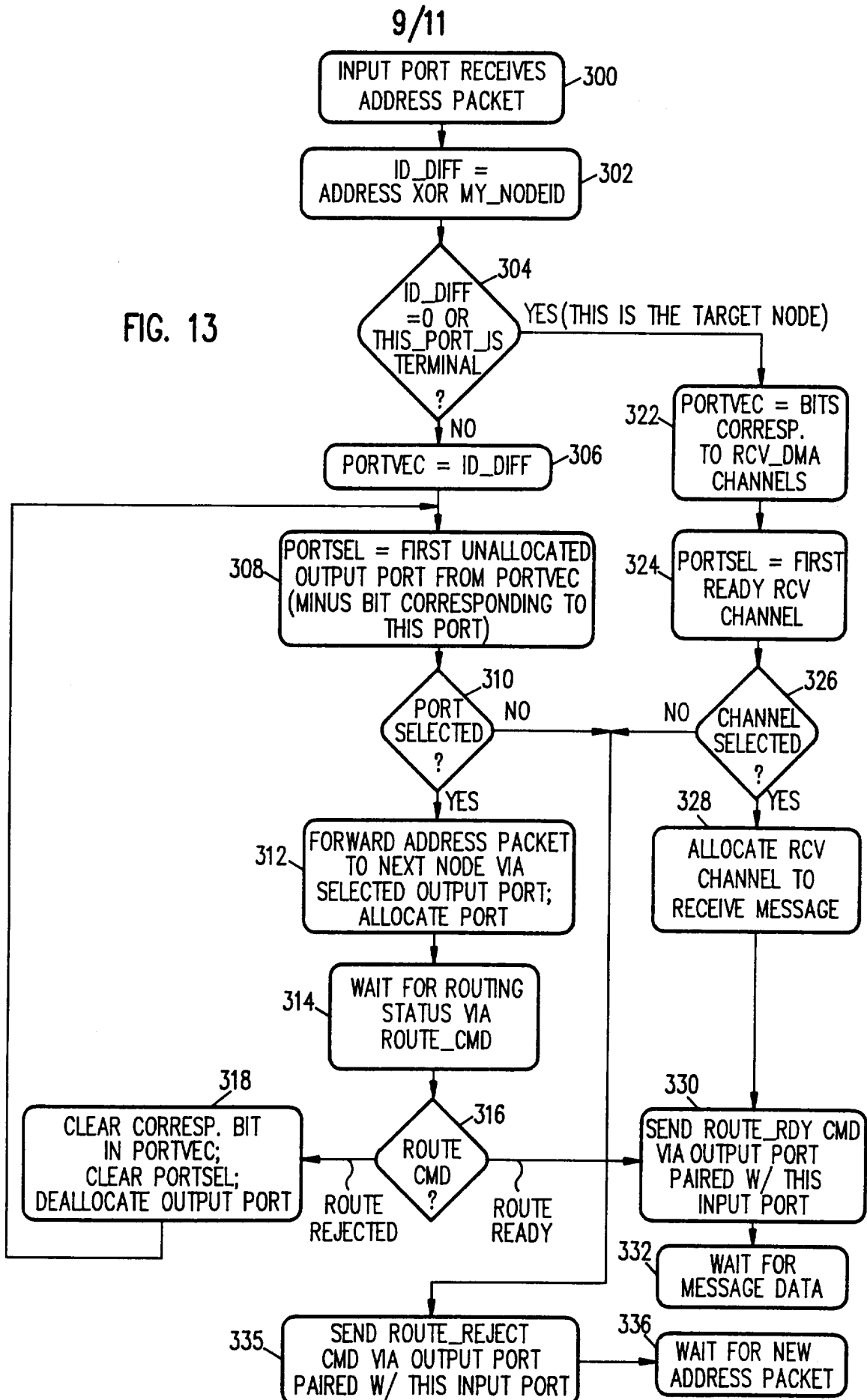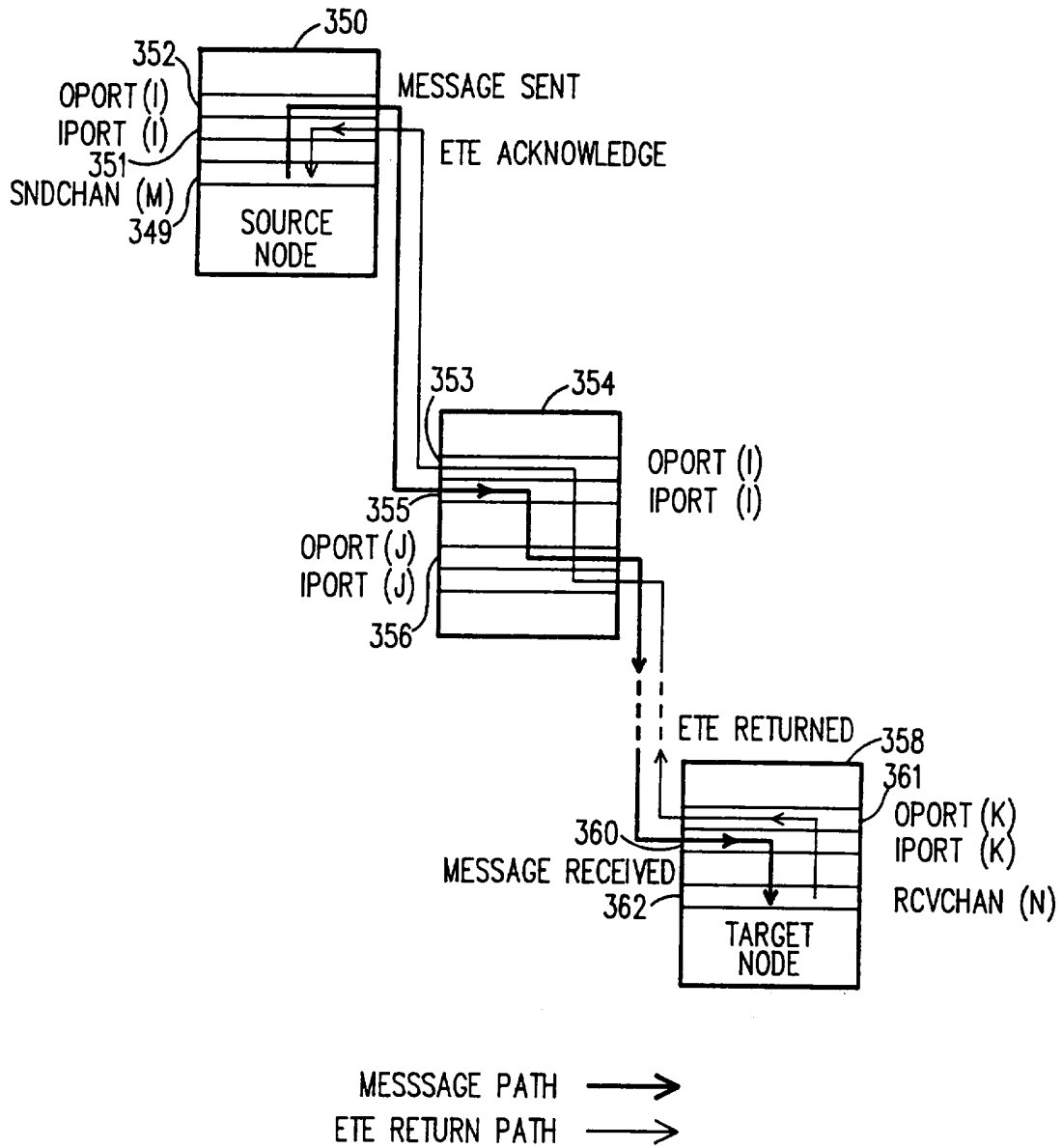ADDRESS PACKET

FIG. 14   END-TO-END ACKNOWLEDGE

11/11

FIG. 15 - MAZE ROUTE TIMING - NO BLOCKED LINKS
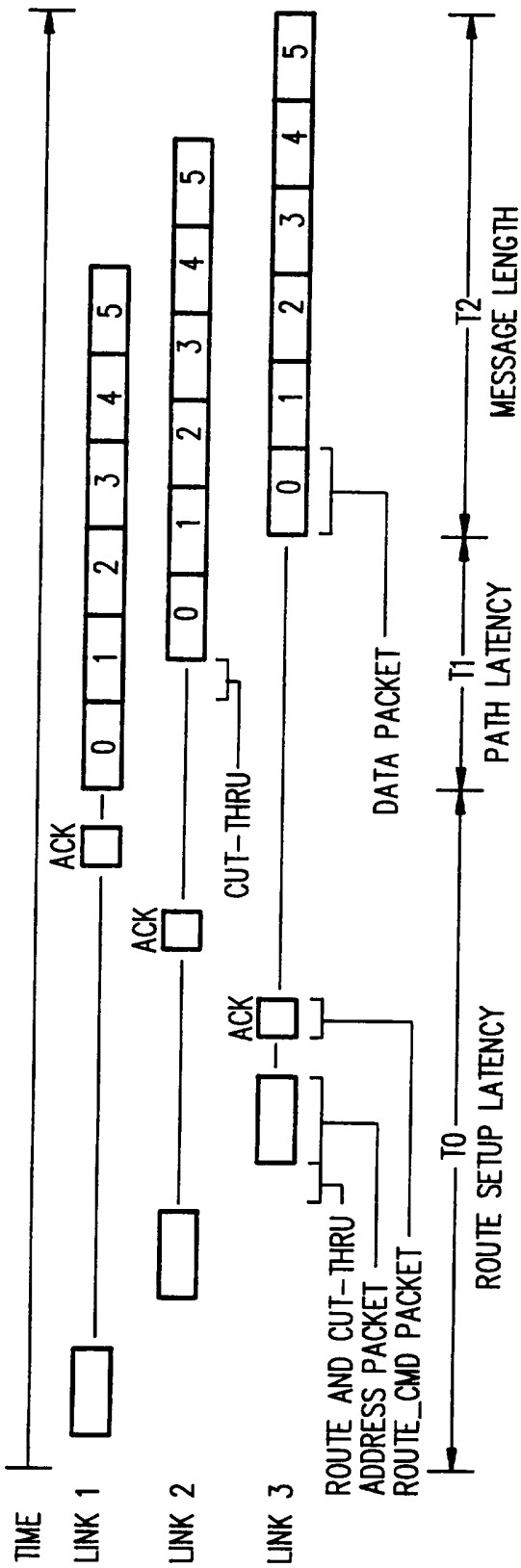
FIG. 16 - MAZE ROUTE TIMING - BLOCKED LINKS AND BACKTRACKING
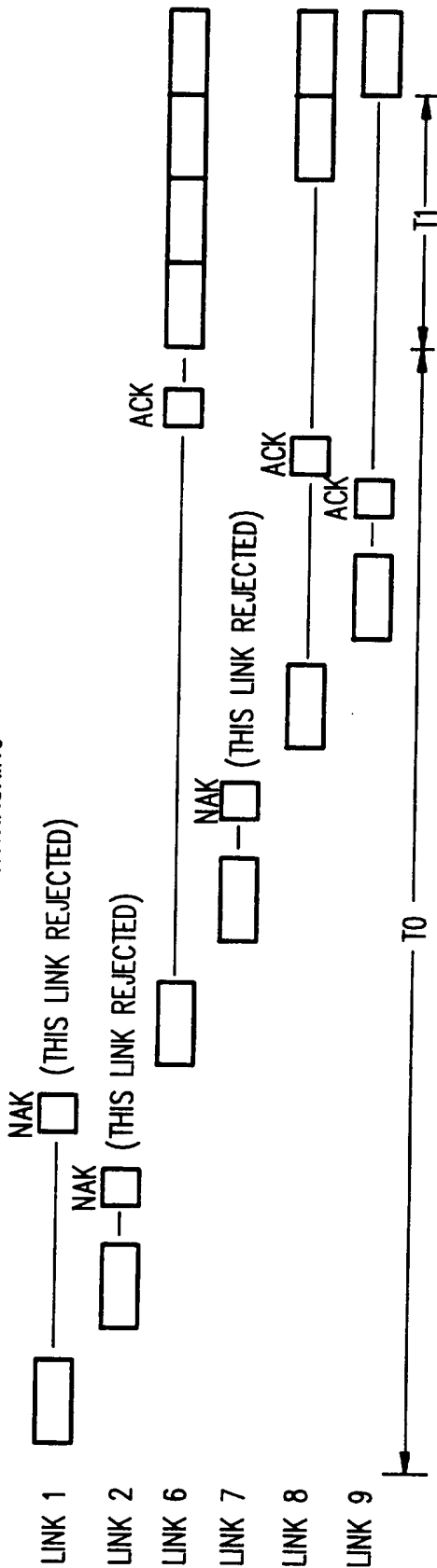
A. CLASSIFICATION OF SUBJECT MATTER
IPC 6   G06F15/16

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6   G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	PARALLEL COMPUTING, vol. 20, no. 4, April 1994 AMSTERDAM NL, pages 509-530, XP 000433524 M. SCHMIDT-VOIGT 'Efficient parallel communication with the nCUBE 2S processor' see page 511, line 3 - page 515; figures 2-5	1,5
A	---	2-4,6-21
Y	COMPUTER ARCHITECTURE NEWS, vol. 19, no. 1, March 1991 NEW YORK US, pages 6-14, XP 000201919 JOHN Y. NGAI 'A framework for adaptative routing in multicomputer networks' see page 7 - page 8	1,5
A	---	2-4,6-21

-/--

| X | Further documents are listed in the continuation of box C. | | Patent family members are listed in annex. |

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 October 1995	1 0. 11. 95

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Soler, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	5TH ANNUAL EUROPEAN COMPUTER CONFERENCE, May 1991 BOLOGNA, pages 353-357, J-L. BÉCHENNEC 'An efficient hardwired router for a 3D mesh interconnection network' see the whole document -----	1-21

1